Ensemble Methods (Kaggle winning entries)

Classifier 1 — KNN
Classifier 2 — SVM
Classifier 3 — DT

$\hat{y}_{KNN}$
$\hat{y}_{SVM}$
$\hat{y}_{DT}$

For an instance, prediction as: $\{$Good, Good, Bad$\}$

Majority voting — Good.

KNN — 20
SVM — 30
DT — 30

Ensemble will predict $\frac{80}{3}$

when does ensemble learning not work well?

(1) Base model is bad

(2) All give similar o/p or prediction

Error of each model $= e_y = .3$

$$P(\text{2 models being wrong}) = {}^3C_2 (e_y)^2 (1-e_y)^{3-2}$$

$$+ \; {}^3C_3 (e_y)^3 (1-e_y)^{3-3}$$

$$= .19 < .3$$

Single classifier (e.g. decision tree)

Q: Feed in same data $\xrightarrow{?}$ Different model / prediction?

Bagging (OR BOOTSTRAP AGGREGATION)

KEY IDEA: REDUCE VARIANCE

But HOW??

Think about cross - validation!!

Similar in spirit $\Rightarrow$ Create different datasets.

BUT, we have a single dataset!!

$\boxed{\text{SAMPLE WITH REPLACEMENT}}$

e.g. if dataset is like.

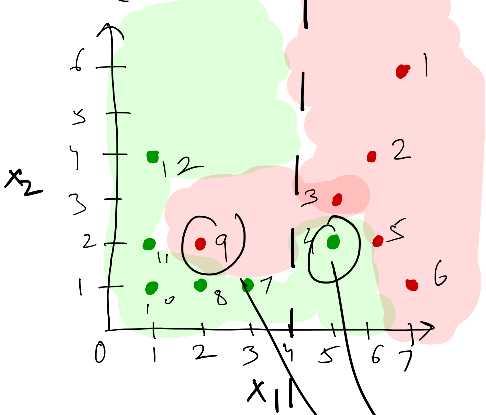
	X		y
Sno	F_1	F_2 . . .	
1			
2			
3			
⋮			
N			

Repetition possible

							N
ROUND	1	D_1	D_3	D_1	D_8	. . .	D_N
ROUND	2	D_2	D_4	D_1	D_{80}	. . .	D_3
	⋮	⋮	⋮	⋮	⋮		⋮

TRAIN SAME CLASSIFIER / REGRESSOR ON THESE DIFFERENT "BAGGING ROUNDS"

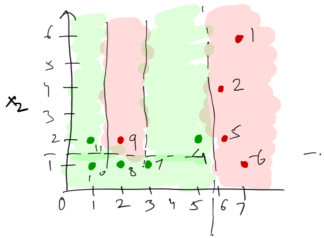
COMPLEX DT (HIGH VARIANCE)



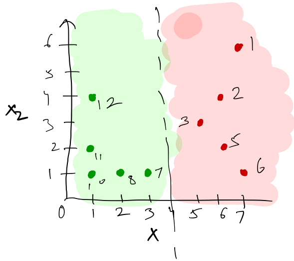
Outliers!

(BAGGING ROUND #1

DT 1 (3,12 missing))



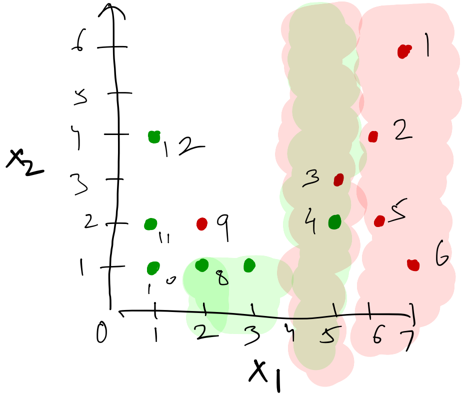
DT 2



BAGGING ROUND #2 (9 and 4 missing)

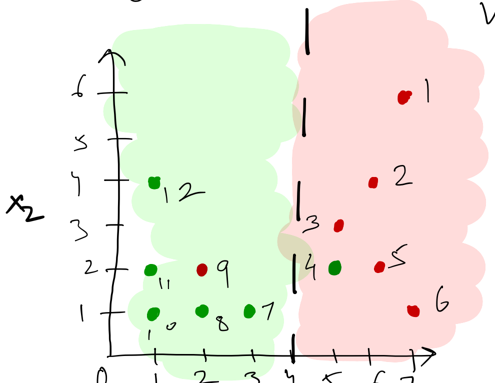
COMBINATION

0 +1



x lesser variance (even with full depth decision trees)

OPTIMAL DT (Reduce variance by decreasing depth)



← we could learn if we limited depth = 1

LABELS AT TEST TIME

300 (+1)

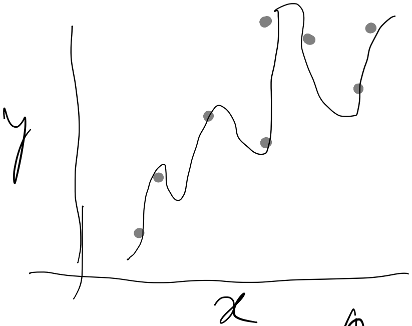
200 (−1)

$$\text{Sign}\big(300(+1) + 200(-1)\big)$$

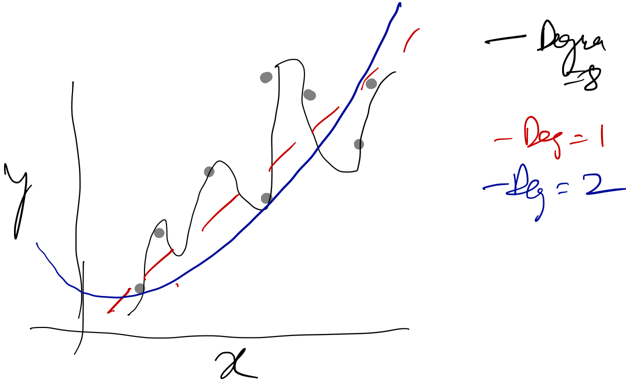
TVe $\xrightarrow{\text{Action}}$ (+1)

“RED”

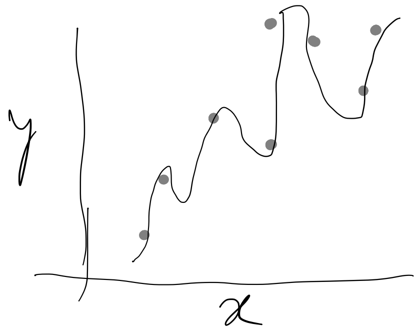
BAGGING FOR REGRESSION



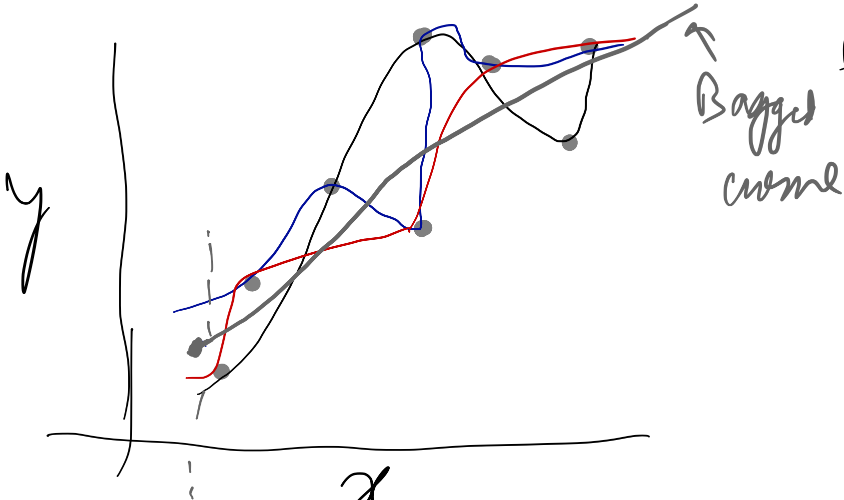
HIGH VARIANCE



Very different if some samples are removed..



ORIGINAL FIT



BAGGING

Bagging

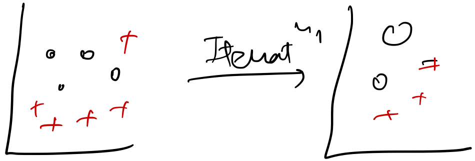
① Taking "strong" learners and combine (to reduce variance)

② Bagging: All learners are independent of each other...

Boosting

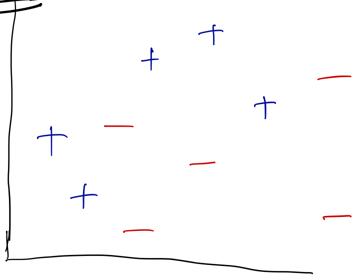
Taking "weak" learners (eg depth-1 D.T.) and combine (to reduce bias)

Learners are incrementally built



"weigh samples differently"

ADABOOST



TRAINING DATA

$w_i \; \forall i = \frac{1}{10} = .1$

ALGORITHM

1) INIT. $w_i^0 = \frac{1}{N}$

2) FOR m=1 to M :

2.1) LEARN CLASSIFIER USING CURRENT WEIGHTS

2.2) COMPUTE WEIGHTED ERROR = $err_m = \frac{\sum w_i (\text{incorrect})}{\sum w_i}$

2.3) COMPUTE $\alpha_m = \frac{1}{2} \log \left(\frac{1 - err_m}{err_m} \right)$

2.4) FOR CORRECT : $w_i \leftarrow w_i e^{-\alpha_m}$

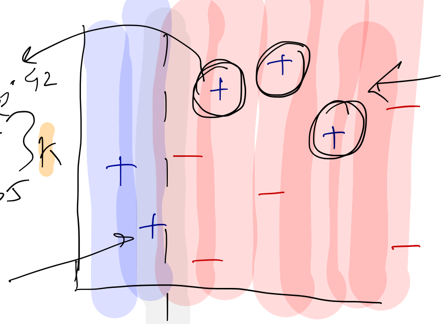
2.5) " WRONG : $w_i \leftarrow w_i e^{\alpha_m}$

2.6) NORMALIZE w_is to SUM TO 1

$w = (.1) e^{.42}$

$= .15 \} K$

$= .165$



Mis-classified.

ITERATION #1

$W_i = w_i e^{-\alpha_1}$

$= (.1) e^{-.42}$

$= .065 \} K$

$= .071$

$err_1 = \frac{\sum w_i \text{ f incorrectly}}{\sum w_i}$

$err_1 = \frac{3 \times .1}{10 \times .1} = .3$

$\alpha_1 = \frac{1}{2} \ln \left(\frac{1 - err_1}{err_1} \right) = \frac{1}{2} \ln \left(\frac{7}{3} \right)$

$= \frac{1}{2} (.85) = .42$

NORMALIZATION $\left(K(.065) * 7 + K(.15) * 3 \right) = 1$

$(.45 + .45) K = 1 \Rightarrow K = 1.1$

ALGORITHM

① INIT. $w_i^0 = \frac{1}{N}$

② FOR m=1 to M :

2.1) LEARN CLASSIFIER USING CURRENT WEIGHTS

2.2) COMPUTE WEIGHTED ERROR = $err_m = \frac{\sum w_i (\text{incorrect})}{\sum w_i}$

2.3) COMPUTE $\alpha_m = \frac{1}{2}\log\left(\frac{1-err_m}{err_m}\right)$

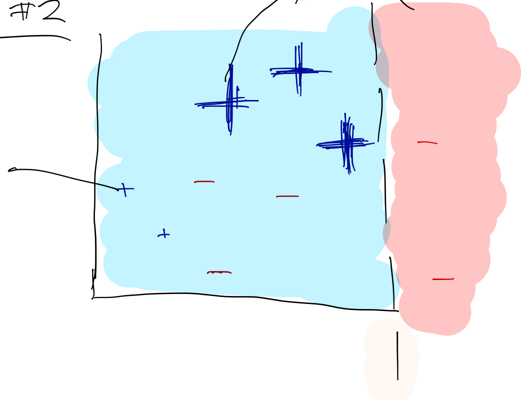
2.4) FOR CORRECT : $w_i \leftarrow w_i e^{-\alpha_m}$

2.5) " WRONG : $w_i \leftarrow w_i e^{\alpha_m}$

2.6) NORMALIZE w_is to SUM TO 1

ITERATION #2

$w_i = \{(0.165) * e^{-0.65}\} Z$



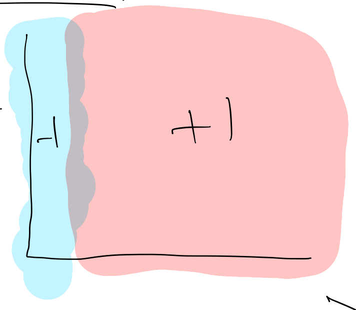
$err_2 = 3 \times 0.071 = 0.21$

$\alpha_2 = \frac{1}{2}\log\left(\frac{.79}{.21}\right) = 0.65$

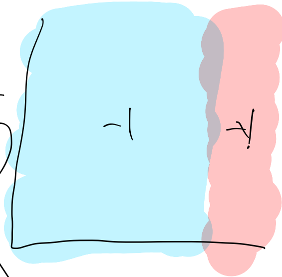
⋮

ADABOOST TESTING

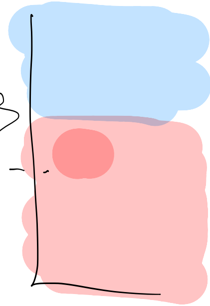
$\alpha_1 \times (0.42)$



$+ \ \alpha_2 \times (0.65)$

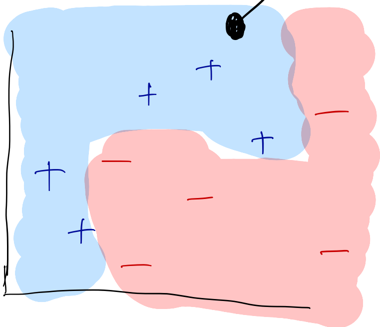


$+ \ \alpha_3$



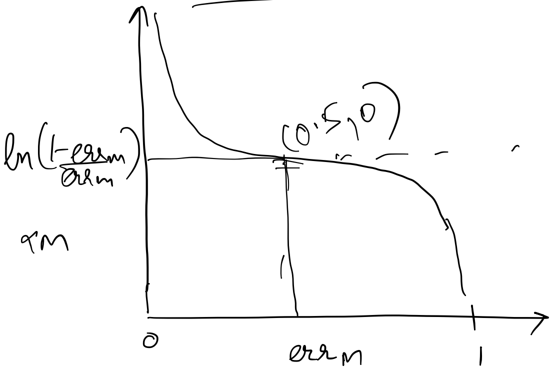
$+ \ \dots$

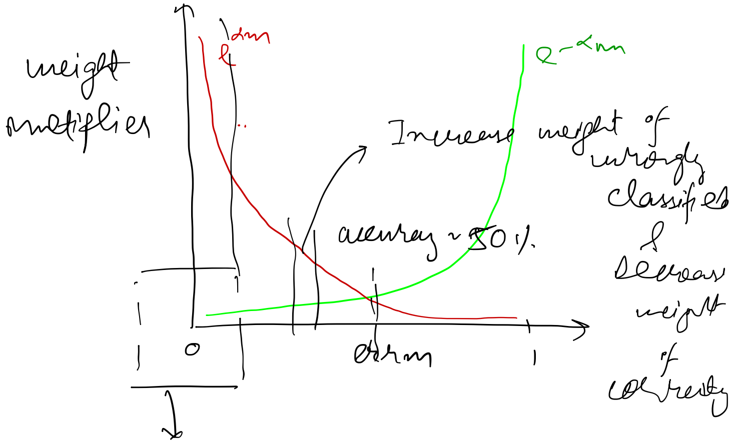
Effective classifier



$$0.42 \times (+1) + (0.65) \times (-1) + \alpha_3 (-1)$$

WEIGHTS INTUITION





$err \to 0$

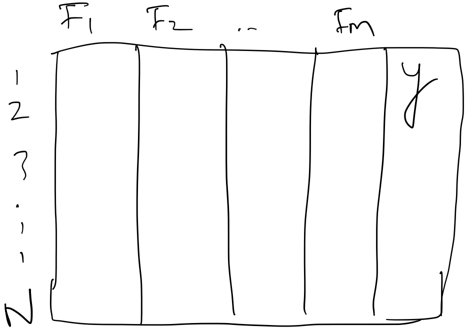
Increase weights of wrongly classified by a large amount.

Random Forests

Reduce variance

$\Downarrow$

Decorrelated trees



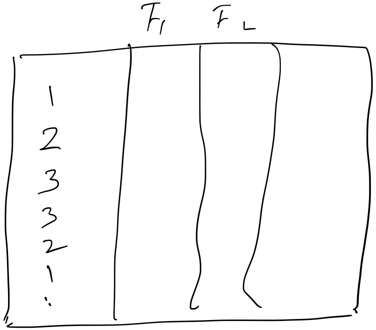
FOR ESTIMATOR m = 1 to n-ESTIMATORS

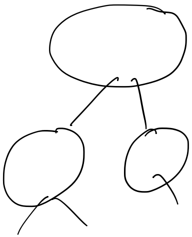
1. Create bootstrap dataset
2. WHILE GROWING TREE

(2.1) CHOOSE m < M features for each node

leads to

(2.2)





F-AVAILABLE = [F_1, F_2 ..]

F-AVAILABLE = [...]

python script.py

depth
tree
feature

1. ~~Depth~~
2. # trees
3. # feature $\{\sqrt{M}, \sqrt{M}+1 \ldots$

$\{1, \ldots M/2 \ldots, M\}$

<1, 100, 1>

<1, 100, 2>

⋮

for depth in [1, ...]:
    for tree in [1, ... ~~M~~ trees]:
        for feature in [......]:
            learnOntrain(depth, tree, feature)